April 2015

Amazon Picking Challenge

Gabriel Thomas Bell
Worcester Polytechnic Institute

Theodore Creighton Armstrong
Worcester Polytechnic Institute

AMAZON PICKING CHALLENGE

A Final Project Report Submitted to the WPI Robotics Engineering Program

Written and submitted by:

Theodore Armstrong
Gabriel Bell

Project Advisor:

Professor Dmitry Berenson

DATE: 30 APRIL 2015

Abstract

The goal of this project was to design and implement the software for a fully autonomous robotic system to compete with in the Amazon Picking Challenge. The Amazon Picking Challenge is a robotics competition with 32 teams from around the globe competing. The challenge is to program a robotic system to autonomously locate requested Amazon products on a shelf, pick the objects up, and place them into an order bin as quickly as possible. Our team at WPI is using a highly precise industrial Yaskawa Motoman robot with two 7-degree-of-freedom arms and a rotating torso. Attached to the robot are Robotiq hands for manipulation and two cameras with color and depth perception to perform object recognition. Our MQP team was responsible for integrating manipulation software with the state machine and object recognition software using the ROS framework. This involved developing a high-level strategy for controlling the robot behavior to complete the task in the fastest possible way. We expect Team WPI to perform well at the competition, which will be held in May at the IEEE International Conference on Robotics and Automation (ICRA) 2015 in Seattle, WA.

Acknowledgements

- Prof Berenson,
- WPI
- Motoman,
- team,
- Amazon,
- Kiva
- Amazon Picking Challenge committee
- Bob (Boisse ?) from ECE shop
- Joe St. Germain

Contents

1 Introduction

Amazon.com, Inc. is the largest e-commerce company in the United States with net sales reaching $89 billion in 2014. Its expanse is massive: having 270 million unique active accounts [1] in over 180 countries [2] 'and routing product orders through 130 active distribution centers [3]. To optimize warehouse logistics, Amazon utilizes a robotic automation solution created by Kiva Systems [4, 5].

Kiva's warehouse system uses mobile robots to transport shelving units around the warehouses [6]. Human workers occupy stationary workstations, and shelves are retrieved by the robots, so the workers can remain in one place instead of walking around the warehouse [6]. Amazon currently utilizes over 15,000 mobile robots in ten of its warehouses [5].

The stationary workstation model makes a good candidate for further automation. However, designing robust and reliable automated picking solutions for unstructured environments remains a challenge. For this reason, Amazon and Kiva Systems have conceived the Amazon Picking Challenge (APC), a competition between 32 international academic and commercial teams to spur the advancement of this technology [7].

The competition challenges teams by presenting a simplified version of the general task of picking items from shelves within an allotted time.--the job a human worker would traditionally do. Successfully completing the task requires utilizing object recognition, pose recognition, grasp planning, compliant manipulation, task planning, task execution, and error detection & recovery [7]. This requires teams to have members with various programming skills and experience. Team WPI has 10 members, a mix of undergraduate and graduate students. The MQP team has been working as a subset within the larger team, spearheading robot manipulation and module integration. Team WPI has been working on the challenge for several months now, getting the system ready for the competition in late May of 2015.

2 Research

2.1 Industrial Robots

Industrial robots are defined in ISO 8373 as an "automatically controlled, reprogrammable, multipurpose, manipulator, programmable in three or more axes which can be either fixed in place or mobile for use in industrial automation applications" with a certain degree of autonomy [8]. The reprogrammable criteria implies that the robot is designed so that any programmed motions or auxiliary functions can be changed without physical alteration of the robot. The robot must be multipurpose in that it must be capable of being adapted to a different application with or without physical alteration. A serial manipulator is a mechanism which consists of a series of jointed segments for the purpose of maneuvering an end effector with several degrees of freedom (see Figure 1 - Serial Manipulator). Importantly, as mentioned earlier, industrial robots must operate with partial autonomy, that is, having the ability to perform intended tasks based on current state and sensing without human intervention. Common applications of an industrial robot include welding, painting, assembly, pick and place, product inspection, product testing, and machine tending. A robotic system is valuable if it is autonomous, robust, accurate, adaptable, accomplishes its task quickly, and is safe [8].

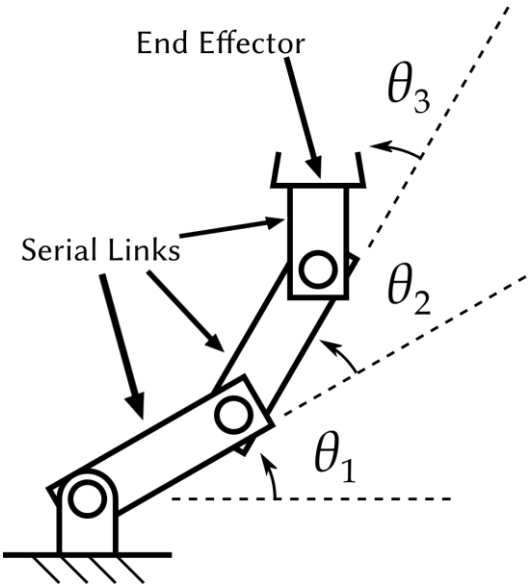

Figure 1 - Serial Manipulator

2.2 Grippers

Robotic systems achieve environment interaction typically through the use of an end effector. Attached to the last link of a serial manipulator is an end effector --a device or tool specifically designed to interact with the robot's environment. Examples of end effectors include a spot welder, a camera, and a prehension device. There are four categories that prehension devices (also known as grippers) fall into: impactive, ingressive, astrictive, and contigutive [9]. Impactive grippers are characterized by grasping the object via physical direct contact to the surface of the object --this is the most common, familiar, and intuitive type of prehension device. Ingressive grippers also utilize direct physical contact, however, they are designed to penetrate the target object (imagine a needle piercing the target object). Instead of pushing into the object to establish and maintain

grip, astrictive grippers utilize a pulling, or suction force applied to objects surface. Such force can be achieved via vacuum suction, magnetism, electroadhesion, etc. The final category, contigutive, achieves its grip by utilizing adhesion via direct contact with the object with glue, surface tension, and freezing. The easiest to implement, and widest available grippers is the impactive physical grasper, the type of which is made available to Team WPI by Robotiq.

2.3 Inverse Kinematics

An industrial robot interacts with its environment by positioning the end effector via actuating its joints, thereby changing the geometry of the arm. Very often, inside each joint, there is a sensor to provide positional feedback to the robot controller. By knowing the angle of each joint, a geometric model can be constructed to determine the position of the end effector --this is known as forward kinematics. In order to command the end effector to move to a desired position, or in a desired motion, inverse kinematics is used. The input is the desired position and orientation of the end effector, and the output is the set of joint angles for the manipulator. The fundamental concept of inverse kinematics is that by working backward from the end effector and applying the law of cosines to two adjacent links at a time, the desired relative angles for both joints are easily obtained. The specific application of inverse kinematics to a given serial manipulator depends on how many joints it has, what the joint limits are, the link lengths, the types of joints (revolute or prismatic), and the configuration of the links. Generally speaking, manipulators with six joints - and therefore six degrees of freedom - have a single joint configuration that will place the end effector at any given position and orientation. Adding a seventh joint provides infinite, continuous joint configurations as solutions to any given desired end effector pose [10]. In the case of having multiple joint configurations as solutions, a single configuration can be chosen based on whatever criteria are desired for the application, such as minimizing overall joint movement.

2.4 Motion Planning

Once a target position and orientation (pose) for an end effector is determined, a new problem arises -- maneuvering the end effector to that pose from the current joint state of the robot. This problem is known as path planning or the piano-movers problem, can be defined by its inputs and outputs. The input being a task description (geometry and physical properties of objects, and spatial relations --either user-provided or sensed), and the output being a sequence of robot motions [11]. This problem is non-trivial when there are objects in the workspace with which the manipulator could collide. That is, there could be solutions for a given pose whereby the configuration of the joints would cause a collision with objects in the environment. Similarly, there could be intermediate configurations between the current robot state and the target state such that the manipulator would collide on the way to its goal. In the context of the Amazon Picking Challenge, it is necessary for our system to minimize collisions at all times. Therefore the robotic system must find a list of intermediary poses from the initial state to the goal state whereby physical constraints (i.e. collision avoidance, speed limits, angle limits, etc.) are respected. Such a problem involves operating within multi-dimensional problem space, and where an exhaustive search is far from ideal -A strong algorithm is required.

Rapidly Exploring Random Tree (RRT) is one of the quickest and most efficient obstacle free path finding algorithm [12]. RRT-star (RRT*), an improved form of RRT, constructs a randomized data structure that is designed for path planning problems [13-15]. This data structure is specifically designed to handle

nonholonomic constraints and high degrees of freedom [14, 15]. The algorithm expands a graph randomly explored in the configuration space, iteratively expanded and guided to the next random state by a set of control inputs [14, 15]. The algorithm is asymptotically optimal implying it would take an infinite time to achieve optimality. Users who utilize the RRT* algorithm for practical planning purposes set a timeout period for which the algorithm would run. When the timeout expires, the algorithm returns the best result achieved. Computing a trajectory 'online' would expectedly return a poor trajectory relative to computing a trajectory 'offline' due to the ability of the offline planner to use a timeout period much greater than the online planner. Therefore, it is advantageous for a robotic system to pre-compute trajectories using the RRT* algorithm offline with a large timeout period for any task configurations known prior to runtime.

2.5 Object Recognition

Robots that operate in unstructured environments need the ability to perceive the world. Introducing image data allows opportunities for object recognition, visual servoing, and robot calibration. To extract meaningful information from two-dimensional (2D) images, powerful algorithms need to be leveraged. Fortunately, Bradski created a computer vision suite, commonly used in applications worldwide [16, 17].Recent technology has allowed for high-fidelity, high-density depth sensing capabilities to be available for a relatively cheap price --an example being the Kinect sensor. Such sensors result in a set of a data points, typically defined as points in Cartesian space. The data is meant to represent the surface of objects in space from the viewpoint of the sensor. Due to 2D and 3D's sensing utility and value to robotic systems, a large scale, open project for 2D/3D image and point cloud processing, called The Point Cloud Library (PCL) was created by Willow Garage, and supported by many organizations worldwide [18]. "The PCL framework contains numerous state-of-the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. These algorithms can be used, for example, to filter outliers from noisy data, stitch 3D point clouds together, segment relevant parts of a scene, extract key points and compute descriptors to recognize objects in the world based on their geometric appearance, and create surfaces from point clouds and visualize them" [18]. The creators analogize PCL to perception as the Boost libraries are to C++.

Useful algorithms for perception for the challenge include template matching, color matching, point cloud stitching, and the Oriented Bounding Box (OBB). Template matching is an algorithm designed to find a part of an image which match a template image. If the target object to be recognized has been learned into a template, the algorithm can search the 2D RGB image for the learned sub-image, and thereby determine if the target is located within the original image. Color matching is another object recognition algorithm where the target object colors (values from an RGB camera) is learned, and then compared to another image. An object would be recognized if there exists a set of pixels similar in color to the learned object's color. Point cloud stitching is a process intended to result in a more accurate and representative point cloud. The method takes point cloud data from several sources and/or angles and transforms them into a common reference frame for further processing. The OBB algorithm takes a set of point cloud data, and returns a minimum volume box in three dimensional space with a definite position and orientation. Such objects can be used conjunctively to recognize, differentiate, and locate objects, with just a 2D RGB camera, and a 3D depth sensor. Only recently has the 3D depth sensor been used in conjunction with 2D RGB camera data been coupled with object recognition. Combining these sensors can produce a "multi-modal object detector" [19]. Team WPI plans to incorporate these methods into their system for the challenge.

3 Project Solution

3.1 Competition Details

At the competition, setup bullpens will be arranged for teams to prepare their systems, perform practice runs, and demonstrate their solutions to the ICRA audience. For the official scored attempt, or "run", our Team will have an hour of setup time to move the robot into place, calibrate the position of the shelf, and perform any other necessary preparation tasks. Ten minutes before the start of the run, the shelf will be visibly blocked off from the Team so that officials can pseudo-randomly place items in the shelf. To start the run, the shelf will be uncovered, and the Team will be given a USB drive with a JSON file containing the information about the contents of the shelf, as well as which items to pick (the work order), to which our software will read and act on autonomously. The robot must retrieve the items on the work order and place the items in the order box within the allotted time limit of twenty minutes. Points are awarded for successfully picked items and deducted for dropping or damaging items, or for picking items that are not part of the work order.

3.1.1 SETUP

Figure 2 - Populated test shelf

The shelving system is a steel and cardboard structure, similar to the shelving units currently used in Kiva warehouses (see Figure 2 - Populated test shelf). A 3D model of the shelf was provided to teams at the onset of the challenge. Teams have used the model both to perform collision checking when planning trajectories and to filter out points from the point cloud generated when one of the cameras is looking inside the shelf and performing object segmentation and recognition. Although the shelf has two identical faces, only one face is being used for the challenge. Twelve of the bins on the shelf will be used, as shown in the picture below, and each bin will contain at least one item.

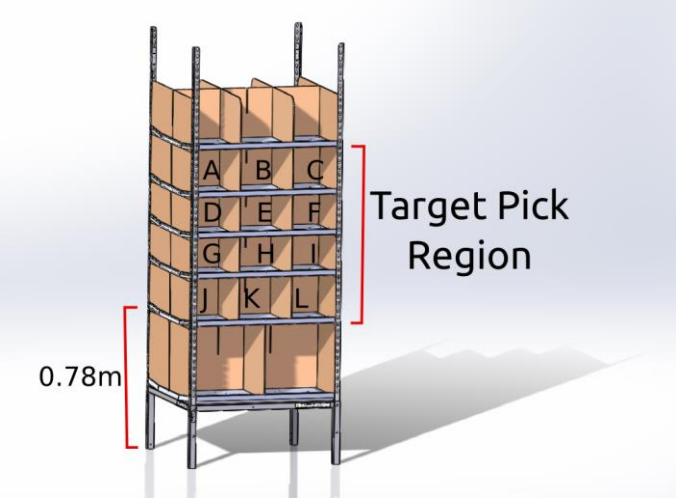

Figure 3 - Model of shelf

In an official attempt, all twenty-five items may be placed in the shelf, or a partial subset of the items may be used. Each team was given the set of items before the competition to conduct practice runs; however, Team WPI purchased an additional set in the event that the items become too damaged to practice with (mainly from dropping items, or crushing them with the grippers). All items on the shelf will be located such that they could be grabbed by a person of average height (170 cm) with one hand. In December, the APC team from Berkeley scanned all the challenge items using their BigBIRD scanning software [20] to create 3D models of all the objects and kindly shared the models with all participants. This data has been invaluable for developing and testing our software to perform both object recognition and grasp planning.

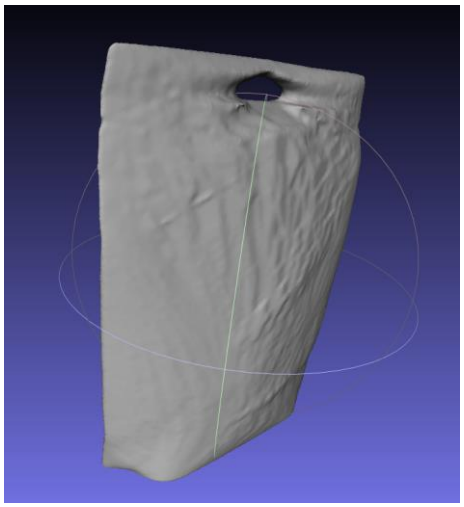

Figure 4 - Cat treats from BigBIRD Scanner

For each team at the competition, the items, their locations, and the work order will change. There will be exactly one item in each bin that is part of the work order. The orientations of items will be pseudo-randomized by the officials stocking the shelf. Items will not occlude one another, from a head-on approach. Additionally, there will be at least two single-item bins, at least two double-item bins, and at least two multi-item bins (three or more items). Duplicate items may be encountered, even in the same bin.

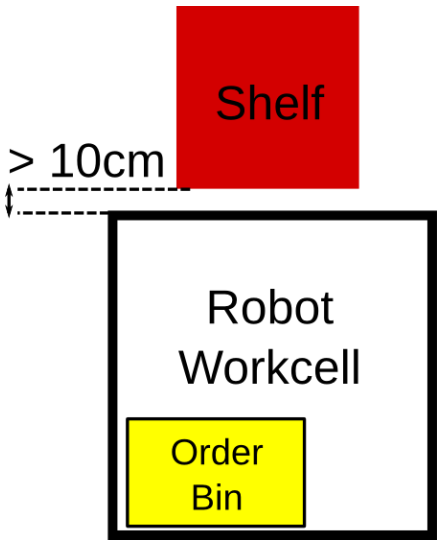

Figure 5 - Workcell layout

The shelf will be placed in a stationary position one hour before the start of the attempt, as shown in the diagram pictured above. Once the shelf and robot are moved into place, our team will calibrate the position of the shelf. Teams are neither allowed to move nor permanently modify the shelf; however teams can place temporary markers on it as long as they are removed prior to the run and don't leave any residual. The entire workcell is a square two meters wide with a gap of at least ten centimeters between the shelf and any part of the robot at the start of the attempt. For safety, teams are required to have an emergency stop button for the system easily accessible. The placement of the order box is up to the team, as long as it is in the workcell, but the robot cannot be supporting the bin at the beginning or end of the run. Moving the order bin outside the workcell will result in zero points for items in the bin.

Two minutes prior to the start of an attempt, we will be given a JSON file with the work order. The file contains a list of the items that must be retrieved, as well as the contents of every bin on the shelf. This means that for performing object detection and recognition for any given bin, the robot will know how many objects are in the bin and what they are. Without this information, the challenge would be far more difficult.

3.1.2 RULES

An attempt is defined as a single scored run. An attempt is over when: the twenty minute time limit expires, the team leader verbally declares the run is complete, or there is human intervention of any kind with the robot or shelf. The items in the work order can be picked in any order; this does not affect the score of the run. Because human intervention is prohibited, there can be no semi-autonomous operation or teleoperation. The number of attempts available to teams will depend on the final number of participants at the competition and the time available, but it will be unlikely to exceed one run.

The scoring rules are shown in Table 1 - Point Distribution. Points will be awarded for placing each item in the work order into the order box. More points are awarded for retrieving items from double-item and multi-item bins, and there will be bonus points (shown in Figure 6 - Pictures of items, and their associated bonus points) for retrieving the more difficult items like the books. Points will be deducted for dropping or damaging items. Teams will also be heavily penalized for picking items that are not part of the work order.

Moving a target item from a multi-item shelf bin into the order bin	**+20 points**
Moving a target item from a double-item shelf bin into the order bin	**+15 points**
Moving a target item from a single-item shelf bin into the order bin	**+10 points**
Target Object Bonus	**+(0 to 3) points**
Moving a non-target item out of a shelf bin (and not replacing it in the same bin)	**-12 points**
Damaging any item or packaging	**-5 points**
Dropping a target item from a height above 0.3 meters	**-3 points**

Table 1 - Point Distribution

Figure 6 - Pictures of items, and their associated bonus points

3.2 System Overview

Team WPI is using a Yaskawa Motoman SDA10F robot as our primary hardware platform, with two Robotiq 3-Finger Adaptive Grippers attached to the arms. There are also two Creative Senz3D cameras attached to the arms to provide depth perception and color images for performing object detection and recognition. Additionally, the Team has constructed a special tray tool that can be used to move or pick up items.

Figure 7 - Overhead of robot system

3.3 Motoman SDA10F

Figure 8 - Yaskawa Motoman SDA10F

The Motoman SDA10F industrial robot is a dual-arm system with seven degrees of freedom in each arm and an actuated torso joint. Having the torso joint in addition to the arms means that Team WPI is able to plan motion paths with eight degrees of freedom for either arm when necessary. The rated payload for each arm is ten kg, which is more than necessary to grasp any object in the challenge. The end effectors on the arms can be controlled with ±0.1 mm precision, which is invaluable for our application, especially when the robot is using the tray to pick up items. The robot can also move at extremely high speeds compared to non-industrial robots. The torso joint, which has the slowest maximum angular velocity, can rotate at a rate of 120°/second. The seventh joint in the arm, which has the fastest maximum velocity, can rotate at a rate of 400°/second. The robot is capable of performing any trajectory within a timespan of several seconds. Therefore it is favorable precompute as many trajectories as possible for the time the system will spend moving the robot (when the robot is not inside the shelf moving slowly) would remain a relatively small percentage of the total time allowed for the challenge.

The robot is controlled by the FS100 controller, also supplied by Yaskawa Motoman. The FS100 is the interface between the manipulators and any external input intending to control the manipulators. The controller performs the necessary computations to operate the robot, including control loops for its actuators, supplying power to the manipulators, and it monitors the status of the robot. Connected to the FS100 controller is a Programming Pendant, used to teach the robot to perform tasks in an industrial setting. For the purposes of the challenge, the Programming Pendant is not used, but the host machine must interface with the FS100 controller in order to maneuver the robot.

3.4 Robotiq 3-Finger Adaptive Grippers

Figure 9 - Robotiq 3-Finger Adaptive Gripper

The Robotiq grippers have three identical underactuated three-joint fingers, meaning that a single motor actuates three joints for each finger. The two fingers on the same side of the palm are each connected to an additional joint so that they can be actuated toward or away from each other. The grippers have four grip modes, as shown in the picture below. In Basic or Wide mode, the fingers remain straight until and unless the first or second finger joints come into contact with an object, at which point the fingers curl around the object. In Pinch mode, the gripper brings the two fingers (on the same side) up against each other and then tries to grab an object with the fingertips. Because the fingers are in direct opposition to each other in this configuration, they obviously cannot curl around an object. Scissor mode is used to pick up very small objects. It is not as powerful as the other modes, but it is useful in certain situations.

Figure 10 - Grasping modes

3.5 System Hierarchy and Component

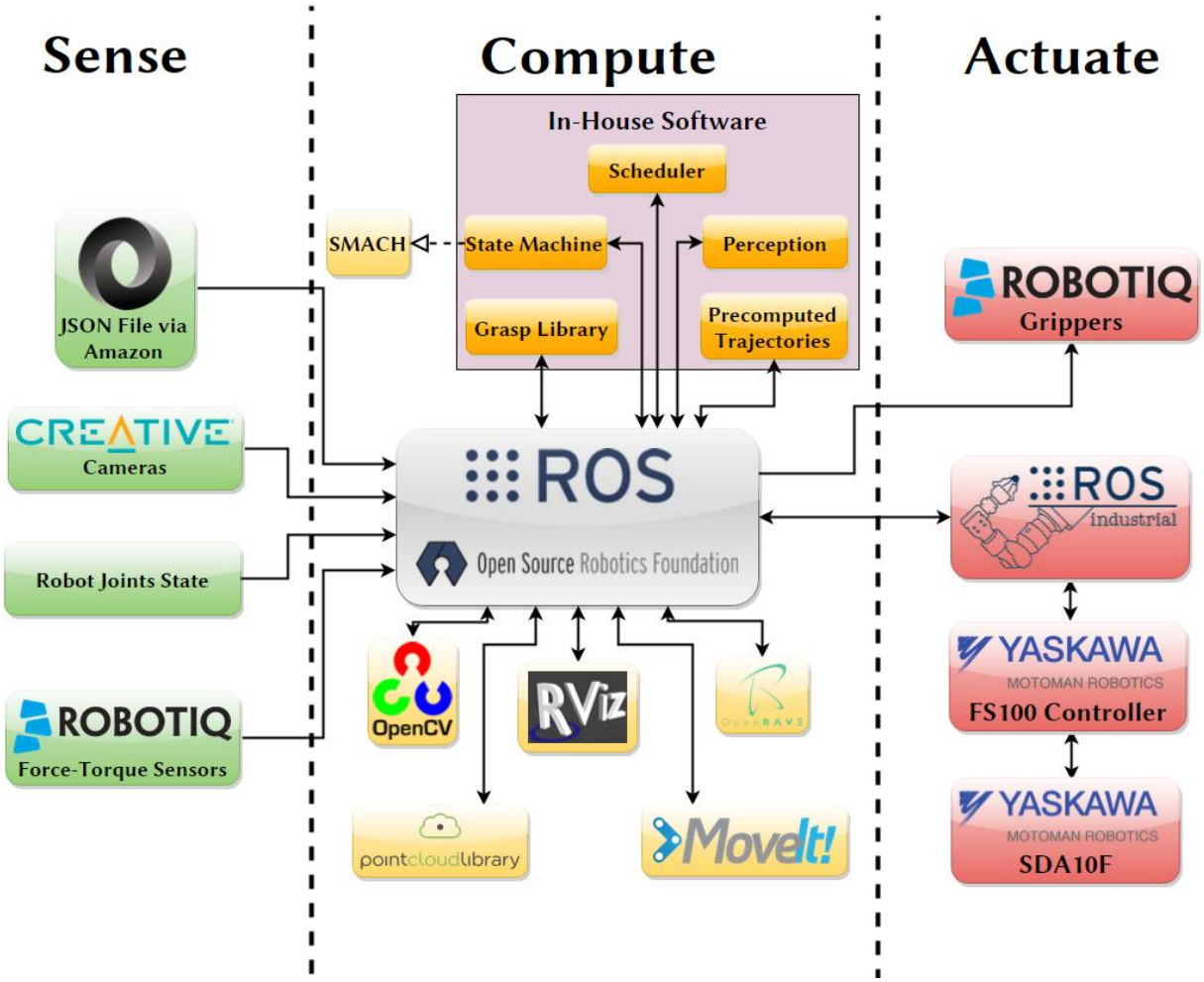

Figure 11 - Software system architecture

3.5.1 ROS

"The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms" [21]. The basic components of ROS are nodes, topics, and services. Nodes are software scripts, written in either C++ or Python (both are compatible with ROS). Nodes communicate with each other using topics. Nodes can publish information to any number of topics, and nodes can subscribe to any number of topics. With this framework, it is very convenient to pass any given node all the information it needs to perform an action. Nodes can also call services. A service is a script that takes an input message and returns an output message. The input and output message formats are specified by the header file for that service. A service could do anything; it may perform an action or it may just return

information. The ROS master node keeps track of all topics, publishers, subscribers, and services in order to enable this inter-nodal communication [21].

In Team WPI's system, ROS controls the interactions between all of the different software modules. The FS100 controller, cameras, and force-torque sensors all stream data back to the computer, and the data is published to various ROS topics. Any software created can subscribe to ROS topics to retrieve any needed information. The Robotiq grippers have a ROS interface that is easy to utilize to send commands for grasp mode, grasp force limit, and finger position. To interface to the robot, the FS100 controller has a custom ROS server that accepts normal ROS Industrial messages and converts them into Yaskawa's own language for controlling the robot. To start the task, the software modules are launched, and the system receives the JSON file containing the work order and other information about the contents of the shelf. It passes the file to the scheduler, which generates a plan, and the state machine is started using the plan as input. The state machine then governs the robot behavior according to the schedule.

MoveIt! is a convenient user interface in ROS for developing software that controls robotic serial manipulators [22] - in the Team's case, the arms and torso of the Motoman robot. The MoveIt! API provides methods to quickly set various parameters and access different path planning algorithms, as well as methods to query different pieces of data (like the position of the robot end-effector). It also takes care of setting up the ROS publishers and subscribers, allowing nodes to easily control robot movement and read joint feedback. The Team's system does not necessitate MoveIt!, but it has helped the Team develop and test software more quickly, and is used for certain scenarios (e.g. generating the Cartesian path that the tray follows to scoop items).

ROS Visualization Tool (RVIZ) is a ROS package that visualizes robots and robot data, like point clouds, coordinate systems, object collisions, SLAM grids, and anything else that can be represented visually [23]. RVIZ has a main window that displays all the data and a corresponding selection panel to hide or show any specific information. It also provides a graphic user interface for path planning and execution, which is useful for testing motions with a robot without having to write any software. When the system starts up, RVIZ is launched; models for the robot, grippers, shelf, order box, and tray are loaded into the RVIZ module. The robot model accurately shows the state of the robot in real time, and all the models are used to perform collision checking during path planning. Also displayed are the point clouds and the RGB images from the cameras as well as any useful coordinate frames. One very useful feature of RVIZ is that trajectories can be visualized before execution, allowing the user to verify beforehand if the robot is going to perform the expected motion.

3.5.2 THIRD-PARTY LIBRARIES

"The Point Cloud Library (PCL) is a standalone, large scale, open project for 2D/3D image and point cloud processing" [18]. It contains algorithms for numerous tasks, including filtering, feature estimation, surface reconstruction, and segmentation. To locate items with the 3D data from the cameras, our system uses Point Cloud Library algorithms. The point cloud data is processed to return the positions and orientations of target items so that an attempt can be made to grasp them. When the system is attempting to find an item with the perception software, processing the point cloud data is the first step. The points that represent the shelf are filtered out, which is possible because our system has a Computer-Aided Design (CAD) model of the shelf and knows where the shelf and the camera both are in relation to the robot. The remaining unfiltered points represent the items in the bin. These remaining points are then segmented into separate objects. If the items

are on the tray, the process is the same, except that the points representing the tray are filtered out instead of the shelf. Again, this is possible because the system has a model of the tray and knows its location.

"OpenCV is an open source computer vision and machine learning software library" [17]. It is used mainly for real-time applications, like facial recognition and movement tracking [16]. Our perception software then utilizes OpenCV to perform object recognition. The library has algorithms available for feature-matching and color-matching, which the Team's system uses to compare the camera images against the stored images of the challenge items. For instance, the box of crayons is a great candidate for feature-matching. All of the faces of the box are colorful and distinct, making the crayons relatively easy to identify. OpenCV also has algorithms for reading text and barcodes which the team may end up utilizing for certain items.

"OpenRAVE provides an environment for testing, developing, and deploying motion planning algorithms in real-world robotics applications. The main focus is on simulation and analysis of kinematic and geometric information related to motion planning" [24]. By giving OpenRAVE the model of the Robotiq gripper and the models of all the items that were generated by Berkeley's BigBIRD scanner [20], potential grasps for any item in the challenge can be pre-computed. The software is mainly used to precompute various grasp positions for each of the items. Whenever an object is successfully located and recognized, our system checks through the precomputed grasps for that item to see if any are possible in the current situation.

3.5.3 STATE MACHINE

We are using a SMACH state machine in our system. "SMACH is a task-level architecture for rapidly creating complex robot behavior" [25] The SMACH library allows users to develop hierarchical state machines. Our state machine is the top level of control in our software architecture. Instead of creating a complex series of service calls and listeners spread throughout the different nodes, the state machine was programmed relatively quickly by explicitly declaring all possible system states and the corresponding outcomes and transitions. For instance, the perception software may fail when attempting to locate a target item, and the state machine knows to pick up the tray as the failure condition, so it then calls the script to pick up the tray. Or, if the target item is located successfully, the state machine knows to call the grasp planner as the success condition, and the system attempts to find a valid grasp for the item. The state machine can also be easily updated at any time to alter system behavior.

3.6 Camera Mounts

For the cameras to provide useful data, they must be securely attached to the robot arms in known locations. This way the referenced coordinate frame for each camera always has a known relation to the robot base frame. To attach the cameras, we designed and created 2 identical 3D-printed mounts, shown below.

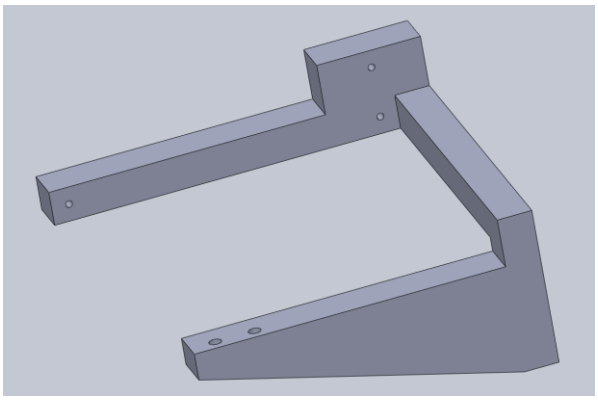

Figure 12 - Computer model of camera mount

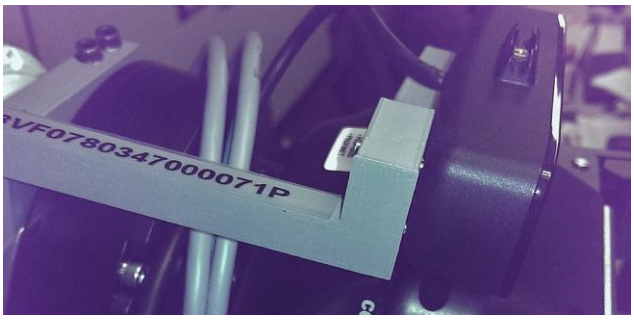

Figure 13 - Mounted camera

3.7 Grasping Items

Our system has two different methods of manipulating items. The primary method is to use the pair of Robotiq grippers which can grasp items in various ways. If possible, our system attempts to use one of the Robotiq grippers to retrieve a target item. However, this relies on both the perception software and grasp planning software to be successful. The perception software must segment the items in the point cloud of a bin, identify the target item, and return its position and orientation accurately. That information is then passed to the grasp planning software. We have precomputed a number of possible grasp positions for each item using the OpenRAVE software. The grasp planner first loads these grasps and their corresponding approach vectors. It then checks the approach vectors in the virtual model and filters out those that collide with the shelf. The software then determines if any of the remaining vectors are possible for the robot to perform with at least one of its hands.

If the OpenRAVE library returns no valid solution, an online grasp planner is called. The online planner takes the segmented point cloud data for the target item and projects the points horizontally onto various planes that are perpendicular to the ground. The plane with the narrowest spread of points is selected, and the approach vector returned is perpendicular to this plane. With this method, the gripper is oriented in the best way to grasp the item, but it is slower than retrieving a precomputed grasp from our library. Like with the offline method, the best potential approach vector is tested to see if the robot can perform the motion and if it

can do it without colliding with the shelf. If that vector fails, the next best potential vector is tested; this is repeated until a vector is returned successfully or every potential vector has failed.

When an approach vector and grasp are successfully returned, the action is performed. Once the item is securely grasped by the gripper, the robot plans and executes a path to get the gripper back to its default pose outside of the bin without colliding with the walls of the bin on the way. From this pose, the robot follows a pre-computed trajectory to the order box and opens the gripper to drop the item into the box.

3.8 Tray

Our alternative method of manipulation is the use of a specially constructed tray (pictured below) that is made specifically to be grasped by the Robotiq grippers and is used to move the items.

Figure 14 - Tray holding items

There are scenarios in which our system can't grasp items with the grippers. Sometimes the perception software does not successfully recognize the item, or it recognizes the item but does not return an accurate bounding box of the item. The point cloud from the camera is very noisy, especially if it is looking almost directly along a planar surface like the side of the Cheez-It box or the cover of a book. Viewing bins from multiple angles helps mitigate this issue, but objects that are very close to walls cannot necessarily be viewed from enough angles to be modeled accurately. Often, the target items are simply in positions in which they cannot be grasped by the grippers at all. For instance, either of the books could be lying flat on the floor of the bin cannot be grasped, and the Oreos basically can't be grasped at all, no matter what orientation they are in.

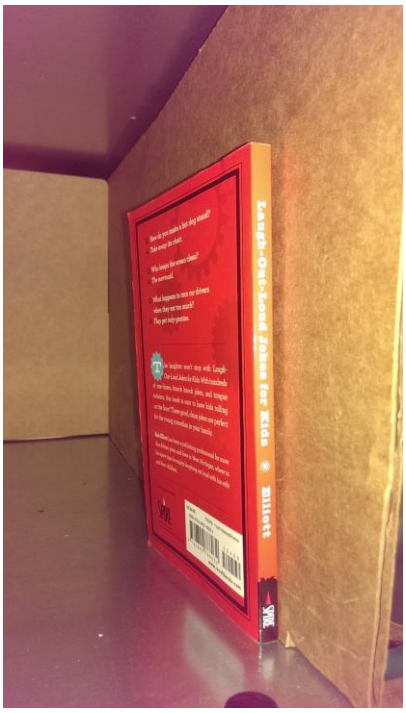

Figure 15 - One of the items in a non-graspable placement

For any such scenario, our system can resort to using our tray to manipulate the items. The tray has two basic uses: it can be slid under items to pick them up, and it can be put into a bin sideways to move the items to one side of the bin when necessary. The tray can also dump items directly into the order box, but we obviously only want to do this for target items.

Figure 16 - Horizontal use of the tray

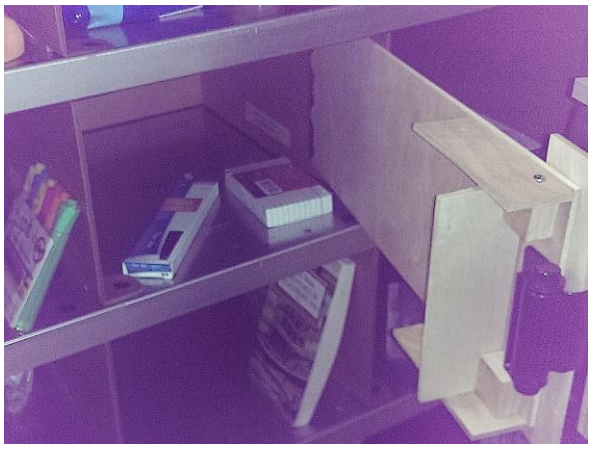

Figure 17 - Vertical use of the tray

3.8.1 TRAY DESIGN

The current version of the tray is constructed mostly from wood. It consists of two rigid wooden parts - which we refer to as the “platform” and the “handle” - connected by a spring-loaded hinge.

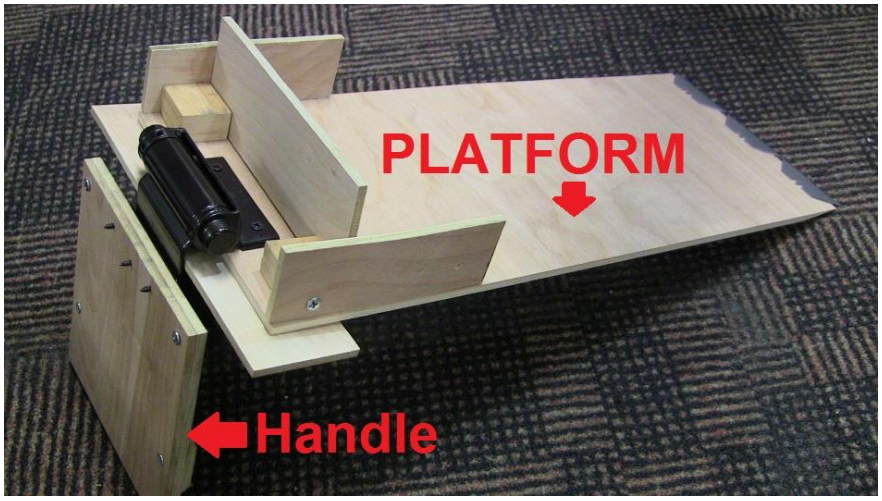

Figure 18 - Tray apparatus

The platform is the part that slides under the items, and a Robotiq gripper grasps the handle to use the tray. Because neither arm can reach all of the bins with the tray, the handle is symmetrical so that either gripper can grasp it.

Figure 19 - Gripper grasping the tray

The tray will rest on a special holder in the workspace when not in use, and it can be reached by both arms to allow for easy transfer of the tray from one gripper to the other. At the front edge of the platform is a line of razor blades that forms a very thin edge about 18 cm wide. The front edge of the wood is also tapered at a very shallow angle so that it can slide under items without lifting the razor edge up off the bottom of the bin.

Figure 20 - Front edge of tray

The short walls toward the back of the platform prevent items from falling off the tray as they slide toward the handle. When scooping items, the items tend to be pushed backward by the edge of the tray until they come into contact with the rear wall of the bin, at which point the tray can continue forward to slide all the way under each item to pick them up.

The Motoman robot is extremely precise, but our knowledge of the position of the shelf is not precise enough to guide a rigid body along its surface with the robot. Instead, the spring-loaded hinge is used to maintain contact between the razor blades and the shelf surface. When there is no force exerted on the platform, the hinge rests at 90°, with the spring holding the platform up against the handle. When scooping items with the tray, the robot tilts the tray forward and places the edge of the platform inside a bin against the bottom surface of the bin. It then lowers the handle, and the hinge exerts downward force on the platform, pressing the razor blades tightly against the bottom of the bin; this allows the tray to easily slide under any item.

Figure 21 – Spring-hinge

3.8.2 TRAY IMPLEMENTATION

The tray can be used in several ways. It is constructed so that it is narrow enough to fit into the bins vertically. This makes the platform about 7 cm narrower than the narrow bins and about 11 cm narrower than the wide bins. This means if we blindly scoop out a section of the bin, we may get all the items in the bin, or we may get only some or none of the items. This leaves us with options depending on what the state machine wants to do.

If the perception software was successful in returning the position and orientation of a target item, then it means we are using the tray because we cannot grasp the item with the grippers. If the target item is something that can normally be grasped from inside the bins and it is simply too close to one of the walls, the system moves the tray into the bin with the platform part in a vertical position and uses the tray to push the target item away from the wall. The system then rescans the bin and reattempts to find a valid grasp for the item. Sometimes this will still fail, or the target item is something that cannot be easily grasped when inside the bin. In this case, we can position the tray to pick up the target item and also attempt to avoid any other items in the bin as much as possible. Once the tray performs a scooping action, it is positioned in front of the robot so that the camera on the other arm can view any items on the tray from various angles.

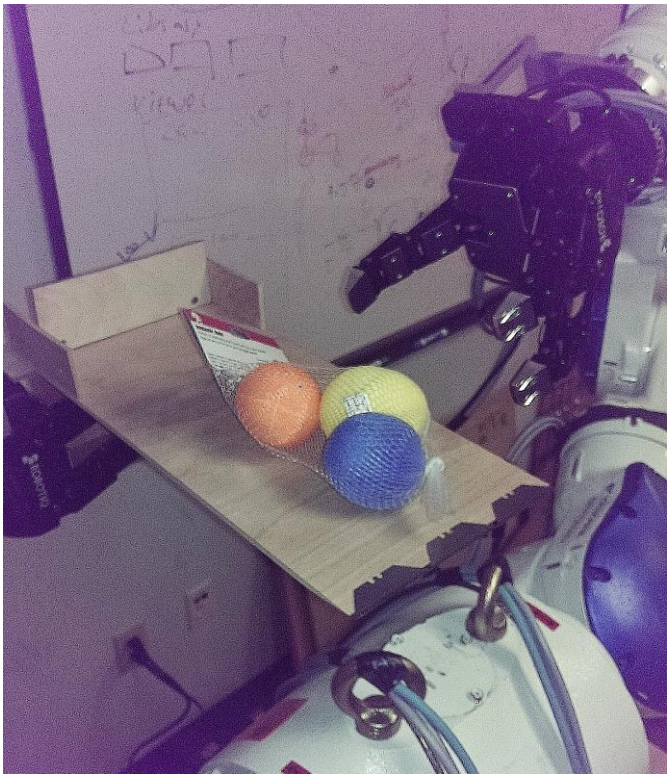

Figure 22 - Perception of items on tray platform

If the perception software determines there is something on the tray and it was scooped from a single-item bin, then it is obviously the target item. In this case, the robot can either use the tray to dump the item into the order box, or, if it is an item that is likely to fall off the tray during movement, the robot can attempt to grab the item off the tray with the gripper and use the gripper to put it in the order box.

If the robot scoops from a double-item or multi-item bin, the system must perform object recognition on all the items on the tray. If there is only one item on the tray and it is the target item, the system follows the same procedure as above. If there are multiple items on the tray, the system attempts to identify which item is the target item. If it can identify and locate the item, the system attempts to find a valid grasp for the item. If there is no valid grasp for the target item (i.e. if it is a book that is lying flat on the tray), the system attempts to find valid grasps for the non-target items so that it may remove them from the tray and place them back into the bin and then dump the target item into the order box with the tray. Alternatively, if there is no valid grasp for the target item and no valid grasp for at least one non-target item, the system can attempt to hold the target item in place with the gripper while dumping the rest of the items back into the bin. If that fails, or if the motion would be impossible to execute, the robot dumps the items back into the bin and either retries that bin or moves onto a different bin after calling the scheduler to determine the next action.

The system can sometimes encounter the case of the tray only scooping items from a bin that are not the target item. The rules state that any item not in the work order must be replaced into the bin they came from or we will be penalized. This leaves the system with three options, which it chooses from depending on the state of the shelf and the items involved: 1) While the robot is holding onto the non-target item(s) with the tray, the system can rescan the bin and attempt to grasp the target item with the gripper; this is likely not possible, since the system would have probably tried to use the gripper (and not the tray) in the first place. 2)

It can dump the item(s) back into the bin and try again to scoop the target item out; this is the preferred solution if the target item is something that can be easily recognized and easily grasped while it is lying on the tray. 3) If there is already an empty bin in the shelf, the system can dump the non-target item(s) into the empty bin and proceed to try again to scoop the target item. However, with this solution, the robot must replace the non-target item(s) back into their original bin or we will be penalized 12 points for each item not replaced.

3.9 Calibration

Because we are not allowed to move the shelf, we must calibrate its position with respect to the robot during the setup period at the competition. Alternatively, we could attempt to precisely position the Motoman robot, but the robot is approximately 220 kg and nearly impossible to position with the precision we require for the challenge. This means calibration of the shelf position is necessary.

There are two potential methods of calibration we are testing, and our team is still in the process of determining which method is more accurate and reliable. The first method is to manually jog each robot arm until a specific chosen point on the gripper comes into contact with a specific chosen point near the corner of the shelf. Once each arm is jogged into place, we run a script that reads the current position of each arm and updates the position parameters in the calibration file. This method works well, but has the downside of requiring teleoperation, which is not required for any other portion of the workflow. The other method is to place optical markers on specific chosen locations on the shelf and run a script that uses the cameras to locate the markers and then updates the calibration parameters. This method has the benefit of being autonomous, but it requires placing markers on the shelf by hand and also depends on the accuracy of the calibrated position of the camera with respect to the robot. We will likely use the first method (manually jogging arms into place) for the competition, as there are fewer variables affecting the reliability of the method.

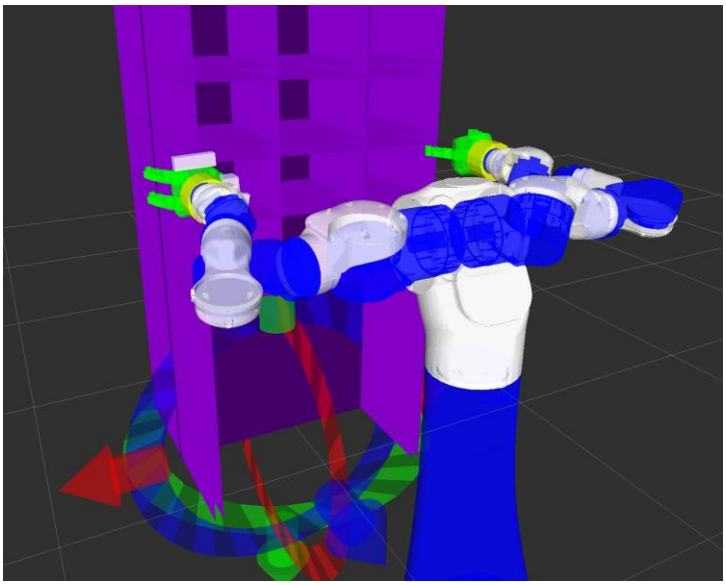

Figure 23 - RVIZ view of shelf calibration

This calibration does not change affect our pre-computed trajectories, as those are intentionally created so that the arms do not move too close to the shelf, allowing for some tolerance in shelf placement. The

calibration does update the model of the shelf in our virtual environment. The motion planning software uses the shelf model to perform collision checking when reaching into the bins with the hands to grasp items. The perception software uses the shelf model to filter the shelf out of the point clouds when searching for items. The calibration also updates the reference frame for planning cartesian paths with the tray. Scooping items (and moving items with the tray sideways) consists of following a simple cartesian path that can be planned very quickly through linear interpolation, with its starting point relative to the current bin. The path must travel along the axes of the shelf coordinate frame (instead of the robot coordinate frame) in order to avoid hitting the walls of the bins.

3.10 Task Workflow

Our ultimate goal is to be able to earn the maximum points possible on every run. The maximum points available will vary depending on how many double-item and multi-item bins there are and what items are in the work order, but it will always be between 150 and 210 points (plus bonus points). To retrieve all the items successfully, every component of our system will need to function properly, and we will need robust error handling implemented into the state machine.

Our system begins a run by scanning each of the bins once, and it attempts to detect and recognize the target item (the item listed in the work order) for each bin. The position and orientation of each target item is stored for the state machine. Multiple camera angles are used for each bin to increase the rate of success and the accuracy of each object's position and orientation. Once all the bins have been scanned, the information is passed to the scheduler. The scheduler determines the best order to retrieve the target items in depending on how much time is left, the probability of success for retrieving each of the target items, and the number of points each target item is worth. It is re-run after each attempt of retrieving a target item. It should be noted that the order in which the items are retrieved does not affect our score.

For each bin, the system default behavior is to attempt to pick up the target item using one of the grippers and drop it into the order box. If the perception software returns failure for a double-item or multi-item bin, or if grasping the item itself fails, the system may rescan the bin and reattempt the grasp before resorting to the tray, depending on the target item involved. When the tray must be used, the system follows the logic described in 3.8.2 – Tray Implementation. Regardless of the manipulation method used, once all twelve items listed in the work order are retrieved and placed in the order box, the run is over, and the system stops. If the system fails to retrieve any items, it will re-attempt each target item until the 20 minute time limit has expired and the run is declared over.

3.11 Results

The system is still being developed by the team to get it ready for the competition. The different software components have all been integrated into the state machine except for the scheduler, which is being saved for last, once all other components are more thoroughly tested. Other than system integration, our main priority as an MQP team was developing the tray and the associated software to the point where the system could rely on the tray to get any item out of the shelf when the perception software or grasping fails.

Most importantly, the current version of the tray is able to pick up any item, including the in the worst case scenario, which is when one of the books is lying down flat and the spine is facing away from the robot. The first two versions of the tray could not pick up the books in that orientation without damaging them. The first version did not have a razor edge and could not easily get under the cover of the book. The second version had a similar razor edge, but the blades were attached to the underside of the tray, meaning that objects also had to then pass over a wooden lip in addition to the razor edge. That tray could slide its razor blades under the book, but not the rest of the tray.

It takes approximately 10 seconds for the robot to pick up or put down the tray with either arm. It takes approximately 18 seconds for the robot to slide under an item and get it out of the shelf. Similarly, it takes about 22 seconds for the robot to go into a shelf vertically and push the contents to one side. The worst-case challenge scenario would involve the need to use the tray both vertically and horizontally, with opposite hands, for each bin, as well as switching hands between each bin. This would mean, over the course of the task, the robot would pick up the tray (and set it down) 24 times, and perform at least 12 horizontal actions and 12 vertical actions with the tray (of course, in a real worst-case scenario, the robot could try and fail forever to scoop just the target item out of a bin). These physical actions add up to approximately 960 seconds, which would take up 80% of the 20 minute time limit.

In the worst-case scenario, the system would likely not complete the task for all twelve target items, but it still might get most of the points available. The worst-case scenario is also extremely unlikely. More reasonably, we expect to use the tray less than half that often in any given run. By our best approximation, the physical tray actions will end up taking around 20-30% of the time available. This leaves plenty of time to perform object recognition and grasp the objects with the Robotiq grippers when possible. It also leaves ample time for the system to handle errors and retry actions. Overall, the current tray is both practical and effective, and the different software components of the system are communicating successfully.

4 Technical Documentation

4.1 Yaskawa Motoman SDA10F

TABLE 2 - BASIC SPECIFICATIONS

Item	**Model**	**MOTOMAN-SDA10F**
Configuration		Articulated
Degree of Freedom		7 axes for the left arm (R1);
		7 axes for the right arm (R2);
		1 rotation-axis
Payload		10 kg/arm
Repetitive Positioning Accuracy		±0.1 mm
Range of Motion	SDA10F: B1 (B2)-axis (rotation)	-170° - +170°
	S-axis (lifting)	-180° - +180°
	L-axis (lower arm)	-110° - +110°
	E-axis (l'ower arm twist)	-170° - +170°
	U-axis (upper arm)	-135° - +135°
	R-axis (upper arm twist)	-180° - +180°
	B-axis (wrist pitch/yaw)	-110° - +110°
	T-axis (wrist twist)	-180° - +180°
Maximum Speed	SDA10F: B1 (B2)-axis	2.27 rad/s (130°/s)
	S-axis	2.97 rad/s (170°/s)
	L-axis	2.97 rad/s (170°/s)
	E-axis	2.97 rad/s (170°/s)
	U-axis	2.97 rad/s (170°/s)
	R-axis	3.49 rad/s (200°/s)
	B-axis	3.49 rad/s (200°/s)

	T-axis	6.98 rad/s (400°/s)
Allowable Moment	R-axis	31.4 N·m (3.2 kgf·m)
	B-axis	31.4 N·m (3.2 kgf·m)
	T-axis	19.6 N·m (2.0 kgf·m)
Allowable Inertia $(GD^2/4)^3$	R-axis	1.0 kg·m²
	B-axis	1.0 kg·m²
	T-axis	0.4 kg·m²
Mass		220 kg
Ambient Conditions	Temperature	0 to 40° C
	Humidity	20 to 80% RH at constant temperature
	Vibration acceleration	4.9 m/s² (0.5G) or less
	Others	Free from corrosive gas or liquid, or explosive gas or liquid
		Free from water, oil, or dust
		Free from excessive electrical noise (plasma)
Power Capacity		SDA10F:1.5 kVA

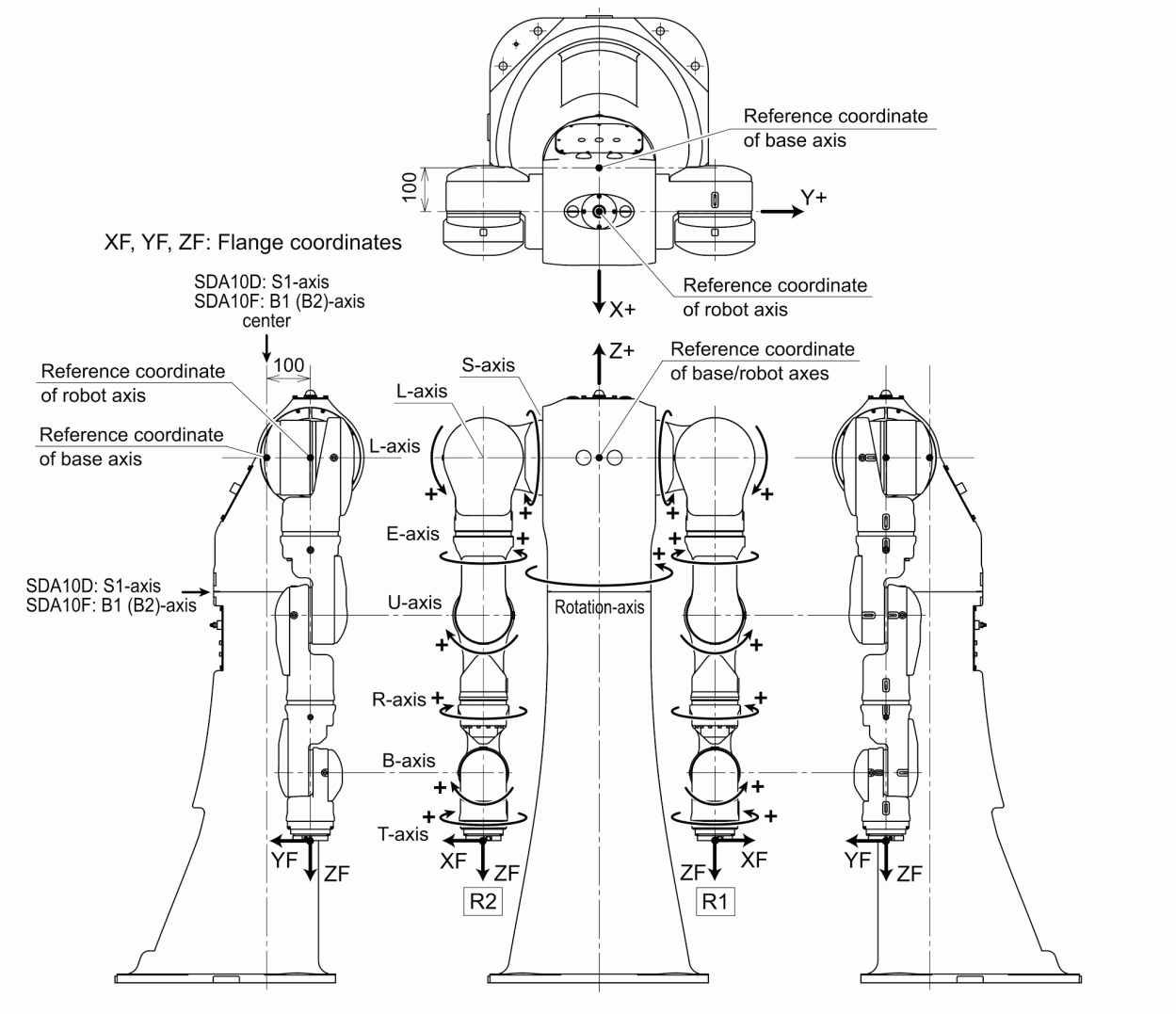

Figure 24 - Part names and working axes

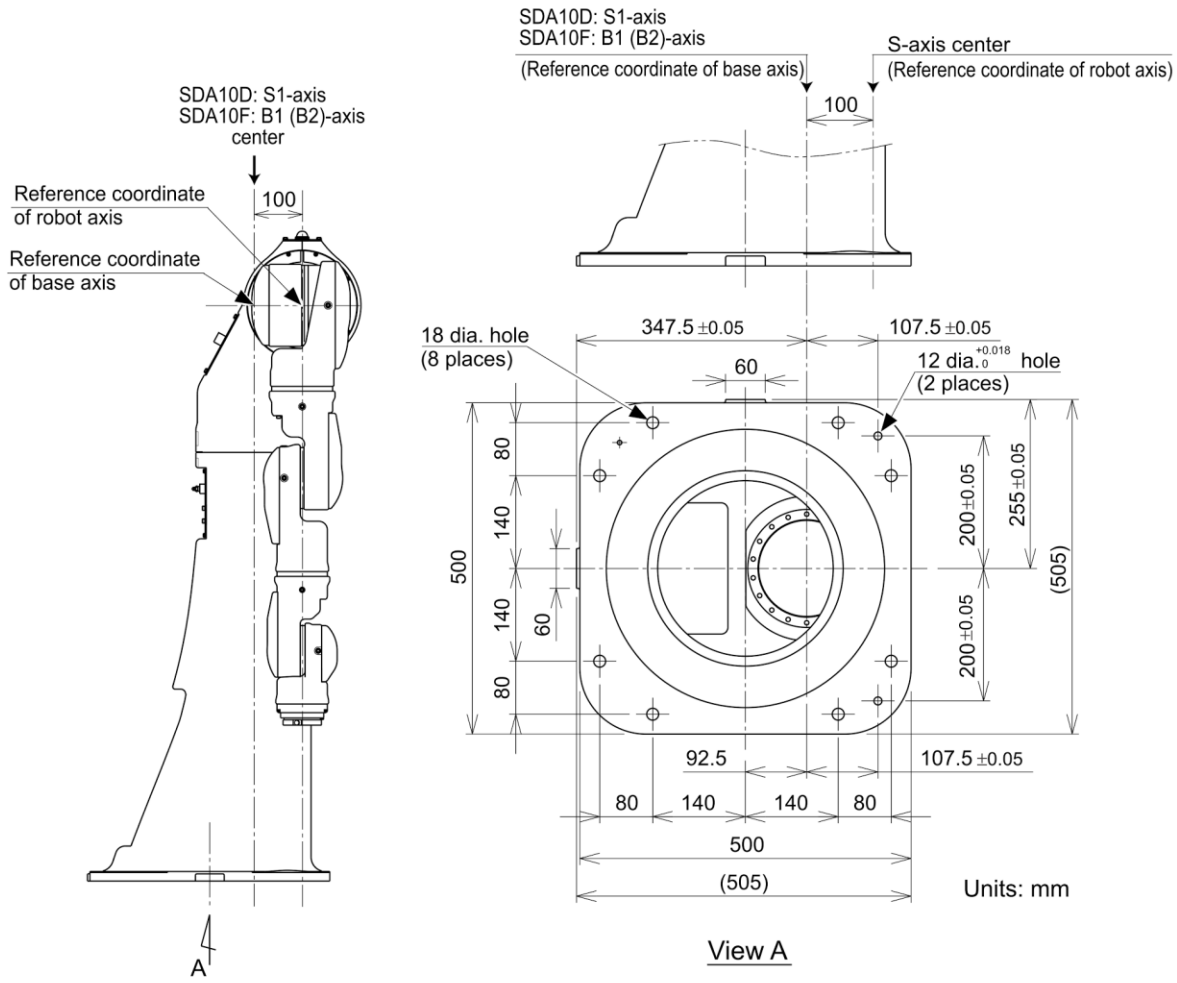

Figure 25 - Manipulator base dimensions

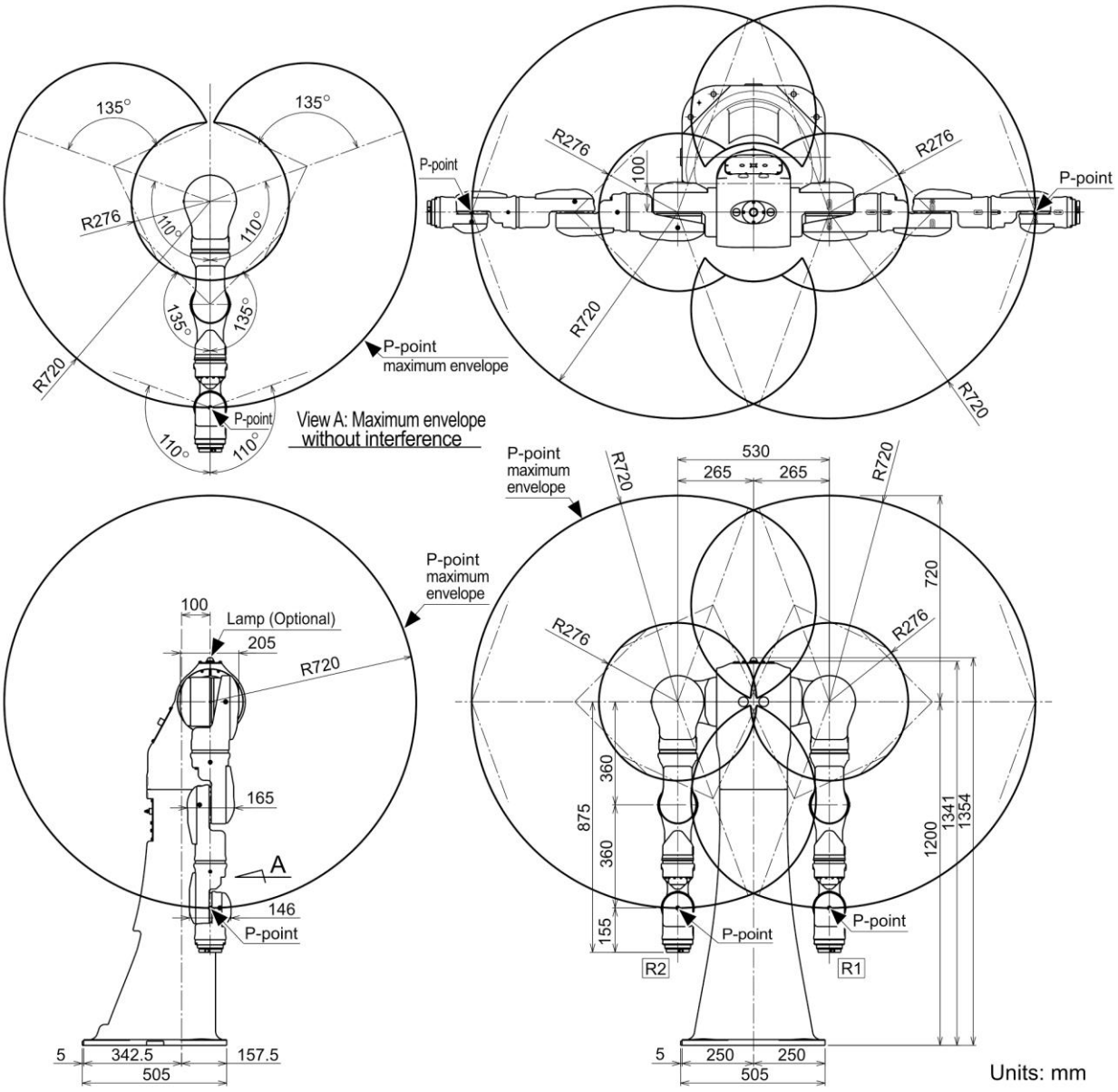

Figure 26 - Dimensions and P-Point maximum

4.2 Robotiq 3-Finger Adaptive Grippers

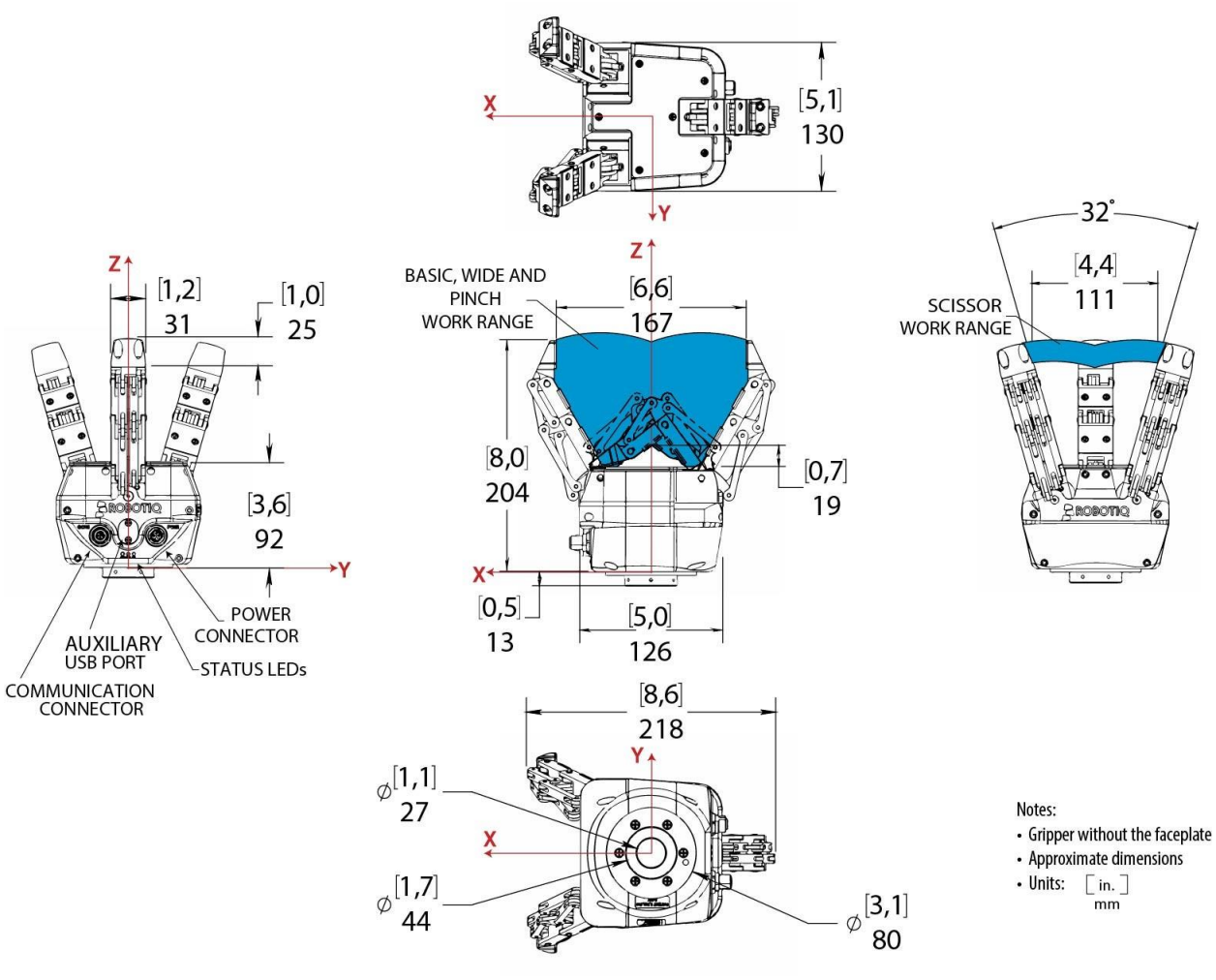

Figure 27 - Technical dimensions

TABLE 3 - MECHANICAL SPECIFICATIONS

Specification	Value
Gripper Opening	0-155 mm
Gripper Approximate Weight	2.3 kg
Recommended Payload (Encompassing Grip)	10 kg
Recommended Payload (Fingertip Grip)	2.5 kg
Maximum Grip Force (Fingertip Grip)	60 N

Maximum Break Away Force	100 N
Maximum Closing Speed (Fingertip Grip)	110mm/sec

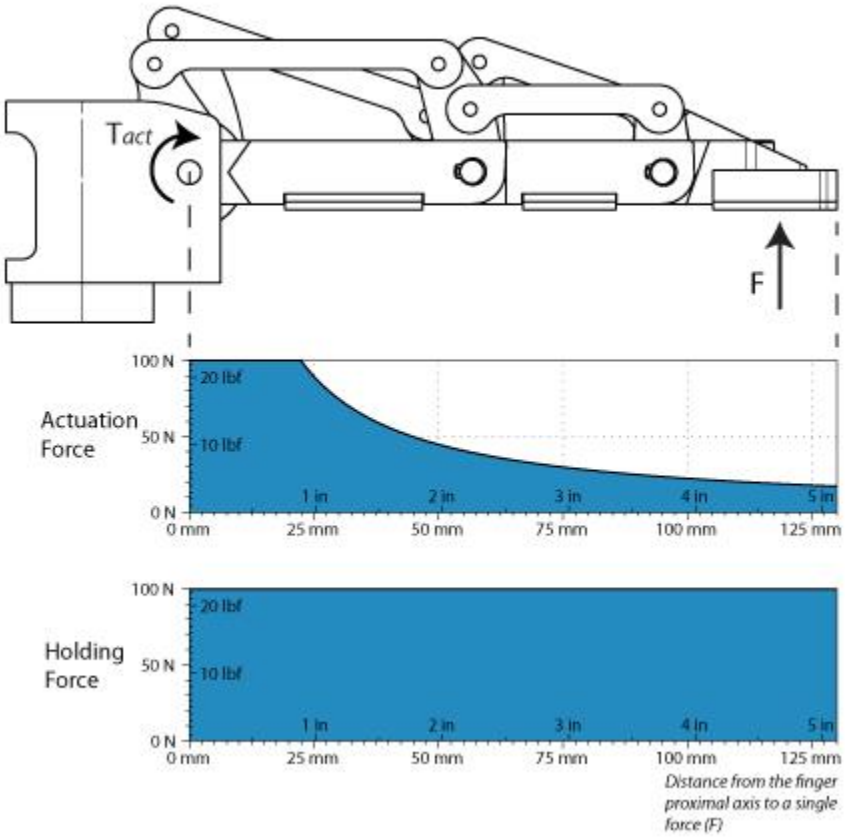

Figure 28 - Finger force graph

TABLE 4 - ELECTRICAL RATINGS

Specification	Value
Operating Supply Voltage	24 V
Absolute Maximum Supply Voltage	28 V
Quiescent Power (minimum power consumption)	4.1 W
Peak Power (at maximum gripping force)	36 W
Maximum RMS Supply Current (supply voltage at 24V)	1.5 A

4.3 Robotiq Force-Torque Sensors

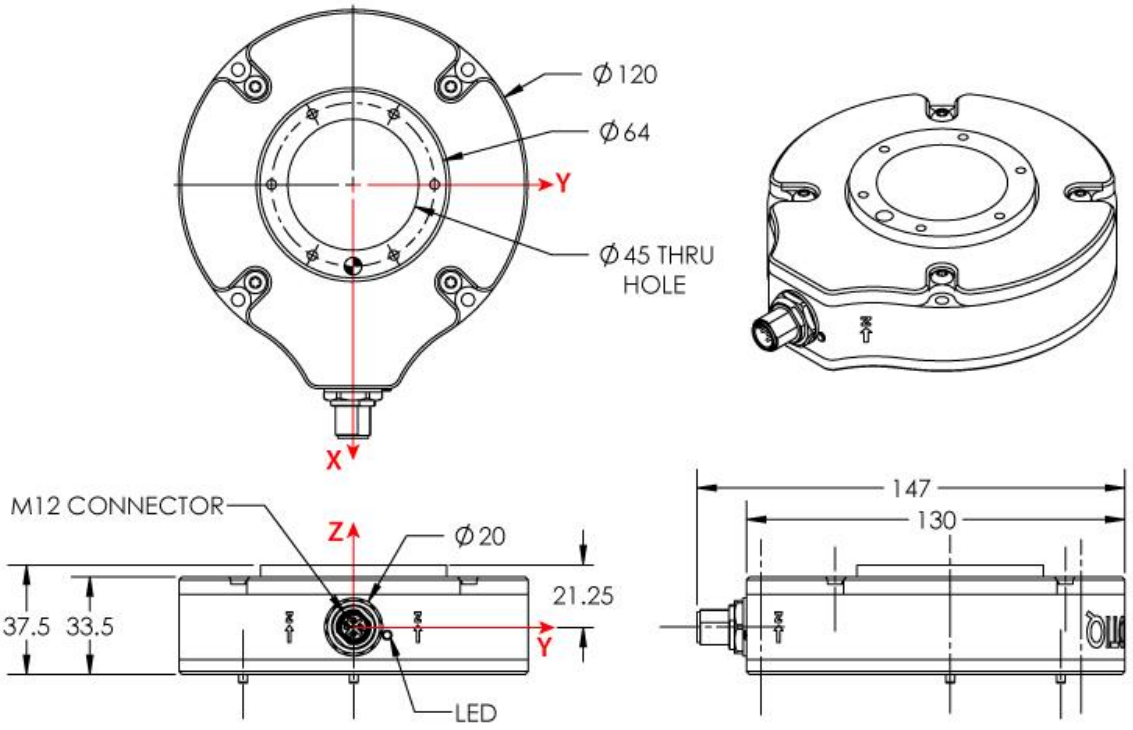

Figure 29 - Technical Dimensions

TABLE 5 - MECHANICAL DIMENSIONS

Specification	Value
Approximate Weight	0.65 kg
Maximum load (all axes)	750 N
Outside diameter	120 mm
Through-hole diameter	45 mm
Thickness	37.5 mm
IP rating	54

Stiffness	F_x, F_y	3.2 X 10^6 N/m
	F_z	3.9 X 10^6 N/m
	M_x, M_y	4700 Nm/rad
	M_z	4600 Nm/rad

TABLE 6 - ELECTRICAL SPECIFICATIONS

Specification	**Value**	
Measuring range	Force	±150 N
	Torque	±15 Nm
Effective resolution	Force	0.2 N
	Torque	0.02 Nm
Signal noise	Combined force	0.5 N
	Combined torque	0.03 Nm
External noise sensitivity	Immune	
Cross-Talking	None	
Accuracy	F_z	1%
	All other	TBD
Drift	Force	±3N over 24h
	Torque	non-significant’
Data output rate	100 Hz	
Temperature compensation	15 to 35 ℃	

TABLE 7 - ELECTRICAL RATINGS

Specification	Value
Input Voltage	6-28 V DC
Max power consumption	2 W
Communication electrical interface	RS-485
Recommended fuse	Phoenix #0916604 (UT6-TMC M 1A)
Recommended power supply	TDK-Lambda DPP Series, *15W Single Output DIN Rail Mount Power Supply*, **DPP15-24**

5 Conclusion

At time of publishing, Team WPI is still hard at work improving the robotic system for the Amazon Picking Challenge. Software development will continue until a scheduled code-freeze on May 9th. Afterwards, the Motoman robot will be shipped to Seattle, WA from WPI on May 14th. If there are no complications with shipping, Team WPI will prepare and then compete in the competition from May 26 to May 28. Further work on the robotic system will focus on increasing robustness of all aspects of the system. The Team expects to increase the system's object recognition rate, increase the total number of grasping strategies, and minimize duty time between movements and states.

We expect the team will perform moderately well in the competition. A majority of teams are keeping their work and designs private, as is Team WPI, and therefore it is hard to compare systems. Our system may not be able to consistently obtain the maximum points in every run by the competition, but it should be able to score at least a decent number of points every run. Depending on how well prepared the other teams are, this may be enough to win. Regardless, Team WPI is excited to compete in the Amazon Picking Challenge with its robotic system featuring the Motoman robot as our hardware platform.

References

[1] Amazon. (January). *Number of worldwide active Amazon customer accounts from 1997 to 2014 (in millions).* Available: http://www.statista.com/statistics/237810/number-of-active-amazon-customer-accounts-worldwide/.
[2] Amazon Press Release. Amazon sellers sold record-setting more than 2 billion items worldwide in 2014. 2015. Available: http://phx.corporate-ir.net/phoenix.zhtml?c=176060&p=irol-newsArticle&ID=2002794.
[3] MWPVL International and Amazon. (April). *Amazon Distribution Network Strategy.* Available: http://www.mwpvl.com/html/amazon_com.html.
[4] Amazon Press Release. Amazon.com to acquire kiva systems, inc. 2012. Available: http://phx.corporate-ir.net/phoenix.zhtml?c=176060&p=irol-newsArticle&ID=1674133.
[5] Amazon Press Release. Amazon unveils its eighth generation fulfillment center. 2015. Available: http://phx.corporate-ir.net/phoenix.zhtml?c=176060&p=irol-newsArticle&ID=1993497.
[6] Kiva Systems. (). *System Overview.* Available: http://www.kivasystems.com/solutions/system-overview/.
[7] J. Romano, P. Wurman, S. Chitta, D. Berenson, D. Pangercic, J. Stückler, Amazon and Kiva Systems. (). *Amazon Picking Challenge.* Available: http://amazonpickingchallenge.org/.
[8] ISO/TC 184/SC 2. Robots and robotic devices — vocabulary. 2012.
[9] G. J. Monkman, S. Hesse, R. Steinmann and H. Schunk, *Robot Grippers.* Wiley, 2007.
[10] C. S. G. Lee, "Robot Arm Kinematics, Dynamics, and Control," *Computer,* vol. 15, pp. 62-80, 1982.
[11] J. Latombe, "Robot Motion Planning," 1996.
[12] F. Islam, J. Nasir, U. Malik, Y. Ayaz and O. Hasan. RRT *;-smart: Rapid convergence implementation of RRT *; towards optimal solution. August 2012, .
[13] Karaman and Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research 30(7),* pp. 846-894. 2011. Available: http://ijr.sagepub.com/cgi/doi/10.1177/0278364911406761.
[14] S. M. LaValle, "Rapidly-Exploring Random Trees A Ш ew Tool for Path Planning," 1998.
[15] S. M. LaValle and J. J. Kuffner Jr, "Rapidly-exploring random trees: Progress and prospects," 2000.
[16] Pulli, Baksheev, Kornyakov and Eruhimov. Real-time computer vision with OpenCV. *Communications of the ACM 55(6),* pp. 61. 2012. Available: http://dl.acm.org/citation.cfm?doid=2184319http://dl.acm.org/citation.cfm?doid=2184319.2184337
[17] G. Bradski. The OpenCV library. *Dr. Dobb's Journal 25(11),* pp. 120-125. 2000.
[18] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," *Robotics and Automation (ICRA), 2011 IEEE International Conference On,* pp. 1-4, 2011.
[19] S. Gould, P. Baumstarck, M. Quigley, A. Y. Ng and D. Koller, "Integrating visual and range data for robotic object detection," 2008.
[20] A. Singh, J. Sha, K. S. Narayan, T. Achim and P. Abbeel, "BigBIRD: A large-scale 3D database of object instances," in *Robotics and Automation (ICRA), 2014 IEEE International Conference On,* 2014, pp. 509-516.
[21] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng. ROS: An open-source robot operating system. 2009, .
[22] S. Chitta, I. Sucan and S. Cousins, "MoveIt! [ROS Topics]," *Robotics & Automation Magazine, IEEE,* vol. 19, pp. 18-19, 2012.
[23] K. Baizid, "ROS Visualization Tool," 2012.
[24] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," vol. 79, 2008.
[25] J. Bohren and S. Cousins, "The SMACH High-Level Executive [ROS News]," *Robotics & Automation Magazine, IEEE,* vol. 17, pp. 18-20, 2010.